执行步骤代码

```
if [ "${R_createEmptyFile_type}" == 'Y' ]  # 判断是否执行input的步骤
then
    >${local_path_mg}/${R_up_fileName}   ## 追加地址到R_up_fileName就是LIST_u_file，即上传文件名
    FILE_NAME_MG=${R_up_fileName}        ##文件名设置为R_up_fileName
else
    #echo "${LIST_pl1}"
    steps1=("${LIST_pl1//\|/ }")       # 取出流程1的所有步骤，设为步骤数组1
    for step in ${steps1[@]}          # 对步骤数组1的所有步骤进行运行
    do
        eval file_id_flag=\${FLAG_${file_id}}
          [ "${d_NextFile}" == 'Y' ] && [ "${file_id_flag}" == 'wait' -o "${file_id_flag}" == 'failed' ] && break
        eval step=\"${step}\"
        step_name=${step%%_*}
        step_type=${step#*_}
        [ "${step_name}" == "${step_type}" ] && step_type=''
    #   echo ${step_name} ${step_type}
        [ "${DOWN_empty_flag}" == 'Y' -a "${step_name}" != 'RNAME' ] && continue
        ${step_name} ${step_type}
    done
    step=''
```

- step_name=${step%%_*}:保留第一个“—”前的字符串
- step_type=${step#*_}:保留第一个“—”后的字符串
- 对保留字符串进行判断是否相等，相等则说明没有“—”存在。
- 然后我们开始执行方法，举个例子：

```
DOWNOK|UNZIP|CK|RM_lineBreak|RNAME_BANKTRAN_temp
DOWNOK|UNZIP|CK_head1|RM_lineBreak|RNAME_BANKTRAN_temp oth
```

- 如果我们的step执行了DOWN，则执行 DOWN方法，传入参数为空。
- 如果执行CK_head1，则执行CK方法，传入参数为head1。

到指定方法查看验证一下：发现确实如此

```
#校验文件函数-----------------------------------------
function CK()
{
    for value in ckType
    do
        getParamValue "${value}"
        eval CK_${value}\=\${getParamValue_value}
    done
    C:\all-aic-project\apphome\sbin\fps\function

    [ "${CK_ckType}" == '' ] && CK_ckType=$1
    #[ "${CK_checkHead}" == 'N' ] && CK_ckType=''
    #[ "${CK_checkLine}" == 'N' ] && CK_ckType=''
    log "校验文件${FILE_NAME[@]}"
```

结合配置文件来看：

处理名称	处理的文件名称匹配规则	默认的生成文件名称	处理方式概述（只针对文件内容，不包括加解压缩以及
BANKTXNLIST	BANKTXNLIST*	BANK:BANKTRAN	每行内容前增加机构号
BANKTXNLIST_yk	BANKTXNLIST*	BANK:BANKTRAN	将“|”分隔的文件内容转为定长，然后每行内容前加
BANKTXNLIST_check	BANKTXNLIST*	BANK:BANKTRAN	机构号

```
|CHECK_line|CG_BANKTRAN.pipe_delimited.sh|CG_addOrg
```

分别进入三个方法的执行脚本中：

- FPS_CHECK脚本中：校验文件下载的完整性。

- FPS_CG脚本中：使用了case针对于setup，addOrg、deleteOrg等都定制了属于自己的方法，但都会执行格式转换和结果取出，然后取出第一个“—"后的文件名，将其导入，并且使用这个方法，得到结果。
- BankTRAN.pipe_delimited.sh最后FILE_NAME和result代入使用write_result方法进行文件格式转换。
- CG_addOrg也调用了FPS_CG脚本:使用了sed -i，直接在每一行里输入org_id这个参数。

```
将“|”分隔的文件内容转为定长，然后每行内容前加机构号
```

查看结果发现和我们上面得出的结果一致。

再来看个BTO的执行流程：

```
|CHECK_line|CG_addOrg
|
|MG_BtoCupsTranFlow:BANKTRAN_temp
```

- 前两个和上面那个一样了没什么说的。
- FPS_MG:拼接一个新名出来，然后对传入的两个文件进行判断是否存在，判断完毕后把文件覆盖到新文件名下，然后使用write_result进行文件合并。

```
流水文件增加机构号，然后和BTO生成文件BtoCupsTranFlow合并
```

会发现结果是一致的。

在看个ETC_TXNLIST_MG执行流程：

```
|
|
|MGETC_ETC_TXNLIST_DAY/|CG_ETC_TXNLIST.sh|CG_addOrg
```

- FPS_MGETC：设置新文件名，根据传入的参数设置文件路径，到指定路径下查看文件，把读取到的所有文件名传入到指定路径下如果结果不为0则合并文件。
- 执行格式转换和结果取出，然后取出第一个“—"后的文件名，将其导入，并且使用这个方法，得到结果,最后FILE_NAME和result代入使用write_result方法进行文件格式转换。
- 最后加上机构号。

```
将白天下载下来的ETC文件合并后，再将“|”分隔的文件内容转为定长，最后每行内容前加机构号
```

疑问：

```
                log "提取文件中的${org_id}的数据"
                sed -ni "/^${org_id}/p" ${local_path}/${CG_file}
                result=$?
            ;;
            rm0x01)
                log "替换文件中字符SOH(0x01)为空格"
                OLD_LANG=${LANG}
                LANG=zh_CN.GBK
                sed -i 's/SOH/ /g' ${local_path}/${CG_file}
                LANG=${OLD_LANG}
            ;;
            *) log "执行脚本"
              export SH_up_path=${UP_path}
              export SH_up_ip=${UP_ip}
              export SH_up_user=${UP_user}
              export SH_up_pwd=${UP_pwd}
              export up_transport=${UP_transferType}
                CG_shell=("${CG_type//:/ }")
                [ ${CG_shell[0]:0-3} != '\.sh' ] || write_result ${CG_file} 11 "检查格式转换脚本"
                CG_xxxxx="${CG_shell[@]} ${local_path}/${CG_file}"
                log "执行格式转换脚本 . ${local_fps_file_transfer_path}/${CG_xxxxx}"
. ${local_fps_file_transfer_path}/${CG_xxxxx}
                result=$?
            ;;
        esac
```

```
eval MG_newName\=\${${SERVER_NAME}_${FILE_ID}_newName}
FILE_NAME=${MG_newName}
```